Appendix C: COE Tools

This appendix lists the COE tools available to assist developers in creating, installing, and testing segments. This appendix is not intended to provide an exhaustive discussion of all tool features. Additional features and improvements in the tools may be made before updates to this document are released. Refer to the Developer's Toolkit release notes, the on-line man pages for the tools, or the help page provided by the tools for the latest information.

The tools are designed to run interactively, and accept one or more command line parameters. Most of the tools do not have an associated GUI, so they may also be invoked from a script.

The tools communicate with the outside world in two ways. First, the tools use the `exit` function to set the Unix `status` environment variable. `status` is set to 0 for normal tool completion or -1 if an error occurs. A `status` return value greater than 0 indicates a completion code that is tool specific.

Secondly, the tools use `stdin` and `stdout` and thus support I/O redirection. Redirecting `stdin` allows the tools to receive input from a file or another program, while redirecting `stdout` allows the tools to provide input to other programs. (Redirecting `stdin` is not always convenient. The `-R` command line parameter, see below, allows a tool to read input from a response file instead of `stdin`.) For example, the tool `COEPrompt` displays a message to the user and allows the user to type in a response, and the user's response is written to `stdout`. The following statement illustrates how this can be used to ask the user to enter the name of a file:

```
COEPrompt "Enter Filename" | MyProg
```

Or, the following can be used to write the results to a file:

```
COEPrompt "Enter Filename" > /tmp/tempfile
```

The command line parameters listed below are common to all tools for which they are meaningful:

`-C file`	Read command line parameters from *file*.
`-h,H,?`	Display on-line help describing how to use the tool.
`-p path`	Use *path* to establish the path for subsequent file names.
`-R file`	Use *file* to respond to questions from the tool.

`-v`	Toggle the "verbose" flag. When enabled, print out diagnostic information as the tool executes. The initial state of the flag is disabled.
`-V`	Display the tool's version number.
`-w`	Toggle the "warnings" flag. When enabled, print out warning messages as the tool executes. The initial state of this flag is enabled.

The parameter `-p` can appear multiple times on the command line. For example,

```
tool -p /h/tst -C cmds -p ../tmp -R responses -p /h/tst2 -C cmds
```

uses the command line parameter `-p` to specify the location of three files: `/h/tst/cmds`, `/h/tmp/responses`, and `/h/tst2/cmds`. The default path for files when the parameter `-p` does not appear is tool specific, but is usually `/h`.

C.1 COE Runtime Tools

This section lists the COE tools that are available at run time. These executables are delivered to the operational site and are located underneath the directory `/h/COE/bin`.

C.1.1 COEAskUser

This tool is intended for use in the `PostInstall` script to display a message to the user and have the user click on a "Yes" or "No" button. The syntax is

```
COEAskUser {parameters} msg
```

where *msg* is the prompt to display to the user. Command line parameters `-C`, `-h`, and `-V` are supported along with the following:

`-YN`	Display buttons labeled as Yes and No.
`-TF`	Display buttons labeled as True and False.
`-AC`	Display buttons labeled as Accept and Cancel
`-B b1 b2`	Display buttons labeled with the strings *b1* and *b2* where *b1* represents a "yes" response and *b2* represents a "no" response

`COEAskUser` sets `status` to 0 if the user clicked on the equivalent of the "NO" button, or to 1 if the user clicked on the equivalent of the "YES" button. `status` is set to -1 if an error occurred (e.g., couldn't display the message requested).

C.1.2 COECopyWS

`COECopyWS` is provided as an aid to speed up the site installation process. It allows segments on one workstation, the source workstation, to be copied across the network onto another workstation, the target workstation. This command will cause the target workstation to be configured identically to the source workstation, including ensuring that the kernel COE is installed and configured. Workstation specific items, such as the target workstation's hostname and IP address are unmodified. Disk partitions and other actions normally performed when the bootstrap COE is loaded are not affected on the target workstation.

The source workstation must already have all segments installed that are to be copied to the target. The target workstation must have at least the bootstrap COE installed, and must be reachable across the LAN from the source workstation.

The target workstation must have sufficient disk space to hold all of the segments that are to be transferred. COECopyWS checks to be sure that this is the case, and reports an error message and aborts if it is not.

COECopyWS does not require that the target and source have identical disk partition configurations. However, segments which are grouped together on the same disk partition on the source workstation will be grouped together on the same disk partition on the target workstation.

This command is normally selected from the System Administration menu, but it may also be invoked from the command line. It may be executed only from the source workstation. The syntax is

```
COECopyWS {parameters}target
```

where *target* is the hostname or IP address of the target workstation. Command line parameters -h, -v, -V, and -w are supported.

The status environment variable is set to -1 to indicate that the target workstation is unreachable, 0 to indicate a successful transfer, 1 to indicate that the target workstation does not have sufficient disk space, and 2 for other errors. Error messages and status information are written to stdout as appropriate.

C.1.3 COEFindSeg

This tool returns information about a requested segment. The syntax is

```
COEFindSeg {parameters}
```

Command line parameters supported are -C, -h, -p, and -V in addition to the following:

-n segname	*segname* is the name of the segment to locate as specified in the segment's SegName descriptor.
-d dirname	*dirname* is the directory where the segment is assumed to be located.

Parameters -n and -d may both be specified if desired. If both are given, the directory specified will be searched first for a match. If no match is found, then an exhaustive search will be made for the segment name given.

COEFindSeg sets status and writes the following line to stdout:

```
directory:segname:prefix:type[:attrib]
```

directory is the absolute pathname for where the segment was found. If the segment was not found, the string "Not Found" is returned and no other information. *segname* is the name of the segment, *prefix* is the segment prefix, *type* is the segment type (translated to all uppercase) and *attrib* is the attribute, if any, found in the segment's `SegName` descriptor.

C.1.4 COEFindServer

`COEFindServer` determines the hostname and IP address of the requested server. The syntax is

```
COEFindServer {parameters} server
```

where *server* is the name of the desired server. Command line parameters `-h` and `-V` are supported.

`COEFindServer` sets the `status` environment variable and writes the following line to `stdout`:

```
IP address:hostname
```

if the server is found. If the server is not found, the string "Not Found" is returned. `status` is set to 0 if the server is found, -1 if it is not found, and 1 if a network timeout occurred.

C.1.5 COEInstaller

`COEInstaller` is normally executed by an operator who selects it from a System Administrator menu. It displays a list of variants or segments which may be installed from tape, disk (e.g., a network segment server), or other electronic media. It may also be executed directly from the command line as

```
COEInstaller {parameters}
```

Command line parameters `-C`, `-h`, `-v`, `-V`, and `-w` are supported as well as:

`-s`	Display the GUI for the network segment server rather than the segment installer which is the default.

`status` is set to 0 if all requested segments were installed correctly, or -1 if *any* segment requested was not installed. By default, `COEInstaller` does not write any output to `stdout`. The verbose and warning flags are both disabled unless enabled by the `-v` and `-w` parameters respectively.

C.1.6 COEInstError

`COEInstError` allows a segment to display an error to the user from within a `PreInstall`, `PostInstall`, or `DEINSTALL` script and terminate installation of the segment. `COEInstError` *always* returns a `status` value of -1 and will cause `COEInstaller`, `TestInstall`, and `TestRemove` to halt further processing of the segment.

`COEInstError` must be used carefully. If it is invoked from within `PreInstall` or `PostInstall`, the installation tools will remove the segment because the installation has been unsuccessful. If invoked from within `DEINSTALL`, the installation tools will *not* remove the segment.

The syntax for `COEInstError` is

```
COEInstError {parameters} errmsg
```

where *errmsg* is a string to display. An error window is displayed with the specified message and the user must respond by clicking on the "OK" button. Command line parameters supported are `-h` and `-V`.

C.1.7 COEListDepends

`COEListDepends` is normally selected from a System Administration menu. It may also be executed from a command line. It displays a sorted list of all segments upon which a specified segment depends. The syntax is

```
COEListDepends {parameters}
```

where parameters `-h` and `-V` are supported as well as:

`-n segname` *segname* is the segment requested.

`-o` Write output to `stdout`.

If `-n` is not specified, the user is prompted to select a segment from a list of segments accessible on the machine. `status` is set to -1 if the segment does not exist. If `-o` is given, output is written to `stdout` in addition to being displayed in a window.

If `-n` is specified, it is assumed that the tool is being run from a command line and the tool writes the output to `stdout` only instead of a window. The first line written to `stdout` is an integer indicating the number of responses to the query. The responses matching the query follow with one response per line in the form

```
segment name:version[{:patches}]
```

C.1.8 COEListSegs

`COEListSegs` is normally selected from a System Administration menu. It may also be executed from a command line. It displays a sorted list of all segments that are accessible from the specified machine. The syntax is

```
COEListSegs {parameters}
```

where parameters `-h` and `-V` are supported as well as:

`-C`	Operate in command line mode only.
`-i address`	*address* is the IP address of the requested platform.
`-n hostname`	*hostname* is the hostname of the requested platform.
`-o`	Write output to `stdout`.
`-st`	Sort the output by segment type.
`-sn`	Sort the output by segment name (default).

Command line parameters `-i` and `-n` are mutually exclusive. If neither is specified, the default is the local machine.

If `-C` is specified, the tool runs in command line mode only and all output is to `stdout` only. If `-C` is not specified, the operator is given a choice of selecting the local machine or another machine.

If output is to `stdout`, because `-C` or `-o` is specified, the IP address and hostname are written out, then an integer indicating the number of responses to the query, then the query results themselves. The format is

```
IP address [:hostname]
n
segname:seg type:version:install date[:patches]
```

`status` is set to -1 if the requested machine does not exist or is unreachable, to 0 if the query was successful, or to 1 if the machine is reachable but does not contain any segments.

C.1.9 COELstProfileSegs

`COELstProfileSegs` is normally selected from a System Administration menu. It may also be executed from a command line. It displays a sorted list of all segments accessible from a specified profile. The syntax is

```
COELstProfileSegs {parameters}
```

where parameters `-h` and `-V` are supported as well as:

`-n profile` *profile* is the profile requested.

`-o` Write output to `stdout`.

If `-n` is not specified, the user is prompted to select a profile from a list of profiles available on the machine. `status` is set to -1 if the profile does not exist. If `-o` is given, output is written to `stdout` in addition to being displayed in a window.

If `-n` is specified, it is assumed that the tool is being run from a command line and the tool writes the output to `stdout` only instead of a window. The first line written to `stdout` is an integer indicating the number of responses to the query. The responses matching the query follow with one response per line in the form

```
segname
```

C.1.10 COELstProfileUsrs

`COELstProfileUsrs` is normally selected from a System Administration menu. It may also be executed from a command line. It displays a sorted list of all user login accounts (designated as local or global) that are assigned to a specific profile, where the profile may be global or local. The syntax is

```
COELstProfileUsrs {parameters}
```

where parameters `-h` and `-V` are supported as well as:

`-n profile` *profile* is the profile requested.

`-o` Write output to `stdout`.

If `-n` is not specified, the user is prompted to select a profile from a list of profiles available on the machine. `status` is set to -1 if the profile does not exist.

If -o is given, output is written to stdout in addition to being displayed in a window.

If -n is specified, it is assumed that the tool is being run from a command line and the tool writes the output to stdout only instead of a window. The first line written to stdout is an integer indicating the number of responses to the query. The responses matching the query follow with one response per line in the form

```
login account:LOCAL | GLOBAL
```

C.1.11 COELstSegProfiles

COELstSegProfiles is normally selected from a System Administration menu. It may also be executed from a command line. It displays a list of all profiles sorted by account group which contain a specified segment. The syntax is

```
COELstSegProfiles {parameters}
```

where parameters -h and -V are supported as well as:

-n segname — *segname* is the segment requested.

-o — Write output to stdout.

If -n is not specified, the user is prompted to select a segment from a list of segments accessible from the machine. status is set to -1 if the segment does not exist. If -o is given, output is written to stdout in addition to being displayed in a window.

If -n is specified, it is assumed that the tool is being run from a command line and the tool writes the output to stdout only instead of a window. The first line written to stdout is an integer indicating the number of responses to the query. The responses matching the query follow with one response per line in the form

```
account group:profile:LOCAL | GLOBAL
```

C.1.12 COELstSegUsrs

COELstSegUsrs is normally selected from a System Administration menu. It may also be executed from a command line. It displays a sorted list of all user login accounts (designated as local or global) that have access to a given segment. The syntax is

```
COELstSegUsrs {parameters}
```

where parameters -h and -V are supported as well as:

-n segname *segname* is the segment requested.

-o Write output to stdout.

If -n is not specified, the user is prompted to select a segment from a list of segments accessible from the machine. status is set to -1 if the segment does not exist. If -o is given, output is written to stdout in addition to being displayed in a window.

If -n is specified, it is assumed that the tool is being run from a command line and the tool writes the output to stdout only instead of a window. The first line written to stdout is an integer indicating the number of responses to the query. The responses matching the query follow with one response per line in the form

```
login account:LOCAL | GLOBAL
```

C.1.13 COELstUsrProfiles

COELstUsrProfiles is normally selected from a System Administration menu. It may also be executed from a command line. It displays a list of all profiles sorted by account group that are assigned to a specific user. The syntax is

```
COELstUsrProfiles {parameters}
```

where parameters -h and -V are supported as well as:

-n name *name* is the user login account requested.

-o Write output to stdout.

If -n is not specified, the user is prompted to select a login account from a list of local and global accounts accessible from the machine. status is set to -1 if the account does not exist, or no profiles are defined for the login account. If -o is given, output is written to stdout in addition to being displayed in a window.

If -n is specified, it is assumed that the tool is being run from a command line and the tool writes the output to stdout only instead of a window. The first line written to stdout is an integer indicating the number of responses to the query. The responses matching the query follow with one response per line in the format

```
account group:profile:LOCAL | REMOTE
```

C.1.14 COELstUsrSegs

`COELstUsrSegs` is normally selected from a System Administration menu. It may also be executed from a command line. It displays a sorted list of all segments that are accessible by a specific user. The syntax is

```
COELstUsrSegs {parameters}
```

where parameters `-h` and `-V` are supported as well as:

`-n name` *name* is the user login account requested.

`-o` Write output to `stdout`.

If `-n` is not specified, the user is prompted to select a login account from a list of local and global accounts accessible from the machine. `status` is set to -1 if the account does not exist, or no profiles are defined for the login account. If `-o` is given, output is written to `stdout` in addition to being displayed in a window.

If `-n` is specified, it is assumed that the tool is being run from a command line and the tool writes the output to `stdout` only instead of a window. The first line written to `stdout` is an integer indicating the number of responses to the query. The responses matching the query follow with one response per line in the format

```
segment name:profile
```

C.1.15 COEMsg

This tool is intended to be used by `PreInstall`, `PostInstall`, and `DEINSTALL` to display an informational message to the user. A message is displayed in a window and the user must click on an "OK" button to continue. The syntax is

```
COEMsg {parameters} msg
```

where *msg* is the string to display. The parameters `-h` and `-V` are supported.

`status` is set to -1 if an error occurs (e.g., a window could not be displayed) and 0 otherwise.

C.1.16 COEPrompt

This tools is similar to `COEMsg`, but expects the user to enter a response. The syntax is

```
COEPrompt {parameters} msg
```

where *msg* is the string to display. The parameters `-h` and `-V` are supported in addition to:

`-C width` — *width* is the maximum number of characters the user may type in the response. The default, if not specified, is 40 characters.

The `status` environment variable is set to -1 if an error occurs (e.g., could not create a window) and to 0 otherwise. The user's response is written as a character string to `stdout` in the form

```
n
string
```

where *n* is the number of characters in the *string* which follows. If n is 0, the user entered a null response and no string follows.

C.1.17 COEPromptPsswd

`COEPromptPsswd` is similar to `COEPrompt` in syntax and operation. It is intended to be used in `PreInstall` and `PostInstall` to prompt a user to enter a password. The user's response is not echoed on the screen.

By default, `COEPromptPsswd` will prompt for the password, then prompt the user to enter the password again for confirmation. The flag `-n` may be used to indicate that a confirmation prompt is not desired.

C.1.18 COERmtInstall

`COERmtInstall` is the remote installation tool. It is normally selected from the System Administration menu, but may also be invoked from the command line. The tool operates in either a "push" or a "pull" mode. The syntax is

```
COERmtInstall {parameters} hostname [filename]
```

where command line parameters `-C`, `-h`, `-p`, `-R`, `-v`, `-V`, and `-w` are supported. Additional parameters are mode specific. *hostname* is the IP address or hostname

of the remote machine for push mode, and the hostname or IP address for the repository site for pull mode. *filename* is the name of the segment file to transfer. If `-p` is not specified, then *filename* must be an absolute pathname.

The `status` environment variable is set to -1 to indicate that the machine requested is unreachable, 0 to indicate a successful transfer, 1 to indicate that the segment file specified was not found, 2 to indicate that there was not enough disk space for the transfer, and 3 for other errors. Error messages and `status` information are written out to `stdout` as appropriate.

Push Mode

Push mode operation consists of an operator at a repository site (e.g., DISA Operational Support Facility) sending one or more segments from the repository site to the remote site.

Push mode operation is indicated by the presence of `-i` or `-s`. The two parameters are mutually exclusive, and may not appear with either pull mode or remote deinstall parameters. *filename* must be specified for push mode operation.

`-s`	Send the file across the network and install it on the network installation server only. Do *not* install the segment at the remote site.
`-i`	Send the file to the remote site, install the file on the network installation server, and then launch the `COEInstaller` tool on the remote machine to allow the operator from the sending site to perform an actual installation.

For example, consider installing a segment file (packaged by the `MakeInstall` tool described below) named `SegPkg_1.1.tar` underneath the directory `/home10/ftp/pub/Segs` to a remote machine named `jdeftest.jdef`, but do not launch `COEInstaller` on the remote machine. A valid command is

```
RemoteInst -p /home10/ftp/pub -s jdeftest.jdef Segs/SegPkg_1.1.tar
```

Pull Mode

Pull mode operation consists of an operator at a remote site requesting the transfer of one or more segment install file(s) from the repository site to the remote site. Pull mode operation is indicated by the presence of `-g` or `-gi`. The two parameters are mutually exclusive and may not appear with push mode or remote deinstall parameters. *filename* must be specified for pull mode operation.

`-g`	Send the file across the network and install it on the network segment server. Do not launch `COEInstaller` on the remote machine.
`-gi`	Same as `-g`, except that `COEInstaller` is launched after the segment files requested have been transferred.

If no command line parameters are specified, and *hostname* and *filename* are both omitted, the tool operates in pull mode but with a graphical user interface. A window is displayed to prompt the user for the IP address or hostname of the repository site. A connection is then made to the repository site, if it is reachable, and a list of segment files is displayed for selection.

Remote Deinstallation

`COERmtInstall` also allows a repository site to remotely remove a segment on a machine at the remote site. This is essentially accomplished by launching `COEInstaller` on the remote machine and using it to select the segment(s) to delete. Remote deinstallation is indicated by the presence of the `-l` parameter. It may not be used with push or pull mode parameters.

`COERmtInstall` provides several security measures (encryption, passwords, anonymous ftp, etc.) to protect segment transmission and to prevent unauthorized access to the repository or a remote site. Discussion of such measures is beyond the scope of this document.

C.1.19 COEUpdateHome

`COEUpdateHome` is intended to be invoked from within a segment's `PostInstall` script. It's purpose is to update the home environment variable within a script file to point to where a segment was actually installed. The syntax is

```
COEUpdateHome {parameters} scriptname envvar
```

where *scriptname* is the filename, including the subdirectory, where the script file exists relative to the segment's home directory. *envvar* is the environment variable to update. The command line parameters `-h` and `-V` are supported.

This tool searches the named file for the first place that the environment variable is defined through either `set` or `setenv`, and replaces the definition with the absolute pathname for where the segment was loaded.

For example, assume `COEInstaller` loads `MySeg` underneath `/home4/MySeg`, and the script file `.cshrc.MYSEG` contains the following:

```
setenv MYS_HOME        /h/MySeg
```

Then the statement

```
COEUpdateHome Scripts/.cshrc.MySeg MYS_HOME
```

changes the statement to read

```
setenv MYS_HOME        /home4/MySeg
```

`status` is set to 0 if the tool is successful, -1 if the file specified is not found, and 1 if the file is found but the environment variable specified is not found.

C.2 COE Developer Tools

This section lists the COE tools which are available during development, but are not delivered to operational sites. By default, these executables are located underneath the directory `/h/TOOLS/bin` and are distributed as part of the Developer's Toolkit. These tools are not location sensitive, and may be moved to any directory desired for development.

All of the tools in this section are executed from a command line. None of them have a GUI interface. Unless otherwise indicated, the syntax for each tool is

```
tool {parameters} segdir
```

where *segdir* is the segment home directory. It is assumed to be under `/h` unless modified by the `-p` command line parameter.

C.2.1 CalcSpace

`CalcSpace` computes the space required for the segment specified and updates the `Hardware` descriptor accordingly. Command line parameters supported are `-h`, `-p`, `-v`, `-V`, and `-w`. The segment referenced must *not* be compressed, and must not contain any files that do not belong with the segment (e.g., source code) at runtime.

The `-v` flag enables the verbose flag and causes the tool to print out the space requirements for each top level subdirectory. The warning flag is enabled by default and causes the tool to print a warning message if directories are found that are not expected (e.g., `include`, `src`) or if expected directories are missing for the segment type (e.g., `bin` for software segments). `CalcSpace` does not affect the reserve space request in the `Hardware` descriptor, and does not affect any partitions specified.

The amount of space required is written to `stdout` in K bytes. The `status` environment variable is set to 0 upon completion or -1 if the *segdir* specified does not exist or does not appear to be a segment.

C.2.2 CanInstall

`CanInstall` tests a segment to see if it can be installed. It performs the same tests that `COEInstaller` does at installation time. To be installable, all required segments must already be on the disk, and there must not be any conflicting segments on the disk.

Command line parameters supported are `-h`, `-p`, `-v`, `-V`, and `-w`. `status` is set to 0 if the segment is installable, otherwise `status` is set to -1. An error message is printed to indicate why a segment is not installable.

C.2.3 ChkCompliance

The `ChkCompliance` tool examines a segment to determine its Category 1 compliance level. Some requirements can not be checked automatically (e.g., using only POSIX calls, expected location of X/Motif libraries), so the tool reports the *maximum* possible level of compliance based on the criteria that can be checked automatically.

The command line parameters `-h`, `-p`, `-v`, and `-V` are supported. If `-v` is specified, the tool will list out the levels as they are checked and validated. The following command line parameters are also supported:

`-f` Print out a form for each level showing which criteria are met by the segment.

`-x` Print out a form showing what criteria must be met to achieve the next higher level of compliance.

C.2.4 ConvertSeg

`ConvertSeg` is a tool that examines segment descriptors and converts them to the latest format. To the extent possible, obsolete usage is identified and corrected as part of this process. The original segment descriptor directory is not modified. The output is in a directory created by the tool and called `SegDescrip.NEW`. This directory will be located directly underneath the segment's home directory at the same level as `SegDescrip`. Command line parameters supported are `-h`, `-p`, `-v`, `-V`, and `-w`.

This command is useful in combining individual segment descriptor files into the proper `SegInfo` descriptor. The tool will print a warning if obsolete directories, such as `progs` and `libs`, are encountered.

C.2.5 MakeAttribs

This tool creates the descriptor file `FileAttribs` described in Chapter 5. It recursively traverses every subdirectory beneath the segment home directory and creates a file with lines in the format

```
permits:owner:group:filename
```

At installation time, the installation tools perform the following statements for each entry:

```
chmod permits $INSTALL_DIR/filename
chown owner $INSTALL_DIR/filename
chgrp owner $INSTALL_DIR/filename
```

For security reasons, `MakeAttribs` does *not* include any file owned by root nor any file for which *permits* is greater than 777. A warning is printed for each filename that is rejected. A warning is also printed for each "suspicious" permission setting such as 777 for a file, execute permissions on files in a `data` subdirectory, etc.

The command line parameters supported are `-h`, `-p`, `-v`, `-V`, and `-w`.

C.2.6 MakeInstall

The `MakeInstall` tool writes one or more segments to an installation medium, or packages the segments for distribution over SIPRNET. `MakeInstall` checks to see if `VerifySeg` has been run successfully on each of the segments, and aborts with an error if it has not. `MakeInstall` supports multi-volume tapes, but will not split segments across tapes. It allows segments for different hardware platforms to be on the same installation medium.

The syntax is

```
MakeInstall {parameters} {seg}
```

where *seg* is a list of one or more segments to process. Command line parameters and segment lists may be repeated as required. A command line parameter applies to all succeeding command line arguments until the next parameter is encountered.

All components of an aggregate must be included at the same time. The order in which segments are passed to `MakeInstall` is unimportant because the tool will order them in such as way as to guarantee that the tape will not need to be rewound during installation.

`MakeInstall` requires that segments be available on disk with at least the `SegDescrip` subdirectories in an uncompressed, unencrypted format. Since segments are normally installed in a repository such as CSRS, `MakeInstall` provides an interface to allow segments to be checked out from the repository as required. This is done by invoking a user supplied program and passing it the name of the segment to extract. The supplied program checks the requested segment out of the repository and places it in an agreed upon temporary location. `MakeInstall` automatically deletes the temporary segments when finished with them. Chapter 5 contains a flowchart which explains the process in more detail.

The command line parameters -C, -h, -p, -R, -v, and -V are supported. If -p is not specified, the implied path is /h. The verbose flag is disabled by default while the warnings flag is enabled by default. If the segments to retrieve are in a repository, the flags -Cd and -Co defined below must be specified.

The following command line parameters are also supported:

-Cd exe path	*exe* is the name of an executable which will check segment descriptors out of the repository and place them in the temporary location indicated by the absolute pathname, *path*. The location of the executable is determined by the most recent occurrence of the -p parameter. Temporary segment descriptors are automatically deleted after the segments are processed.
-Co exe path	*exe* is the name of an executable which will check segments out of the repository and place them in the temporary directory indicated by the absolute pathname, *path*. The location of the executable is determined by the most recent occurrence of the -p parameter.
-d on | off	This parameter specifies whether to delete (-d on) or not to delete (-d off) segments checked out of the repository when MakeInstall completes. By default, segments are not deleted from the temporary directory.
-f	Print segment names as they are processed.
-o file	Use the disk as the output medium instead of tape. The file created is compressed and in tar format, but not encrypted, and has the filename *file.Z.tar*. The purpose of this parameter is to package a collection of segments for installation via COERmtInstall. COERmtInstall will encrypt the data prior to transmitting it across the LAN, and decrypt upon receiving it.
-ot {files}	*files* is a list of one or more segment files created by MakeInstall with the -o command line parameter. This parameter writes files to the output medium in a format suitable for installation by COEInstaller.
-s {segs}	*segs* is a list of one or more segments to write to the output medium. The location of the segments is determined by the

most recent occurrence of the -p parameter. These segments are *not* in a repository.

-so {segs} *segs* is a list of one or more segments to write to the output medium. The segments are to be checked out of the repository by the program provided by the -Co and -Cd parameters.

-S {files} *files* is a list of one or more files that contain a list of segments to write to the output medium. The location of the file is determined by the most recent occurrence of the -p parameter, while the segments are either located under /h, or as determined by -p parameters *within* the file. These segments are *not* in a repository.

-So {files} *files* is a list of one or more files that contain a list of segments that are to be checked out of the repository specified by the -Co and -Cd parameters.

-t dev Use *dev* (e.g., /dev/rst0) as the output medium.

-T Read and display the medium table of contents without performing any other actions.

-va {vars} *vars* is a list of one or more variant definitions to write to the output medium. The location of the variant definitions is determined by the most recent occurrence of the -p parameter. These variants are *not* in a repository. All of the segments which comprise the variant must also be included (via -s, -S, -so, or -So) or else an error will be generated.

-vao {vars} *vars* is a list of one or more variant definitions to write to the output medium. The variant definitions are to be checked out of the repository by the program provided by the -Co and -Cd parameters. All of the segments which comprise the variant must also be included (via -s, -S, -so, or -So) or else an error will be generated.

-Va {files} *files* is a list of one or more files that contain a list of variant definitions to write to the output medium. The location of the file is determined by the most recent occurrence of the -p parameter, while the variant definitions are either located under /h, or as determined by -p parameters *within* the file. These variants are *not* in a repository. All of the

segments which comprise the variant must also be included (via -s, -S, -so, or -So) or else an error will be generated.

`-Vao {files}` *files* is a list of one or more files that contain a list of variant definitions to be checked out of the repository specified by the `-Co` and `-Cd` parameters. All of the segments which comprise the variant must also be included (via `-s`, `-S`, `-so`, or `-So`) or else an error will be generated.

`-x` Perform all of the processing required, but do not actually create the output medium. This is useful to validate a collection of segments and variants to make sure they can be written to the output medium.

The `status` environment variable is set to -1 if any errors occur and to 0 otherwise. Error messages are written to `stdout` as appropriate.

For example, assume four segments (`IDE`, `RDA`, `SEWC`, and `ASW`) are to be written to an installation tape. Assume that `IDE` and `RDA` are underneath `/home2/CM`, but assume that `SEWC` and `ASW` are underneath `/home3/local`. The proper command is

```
MakeInstall -p /home2/CM IDE RDA -p /home3/local SEWC ASW
```

As another example, assume the previous segments `IDE` and `RDA` are stored in a repository. Assume that the program `COSeg` will check out the requested segment and store it in the directory `/h/tmp`, and the program `COSegDescrip` will check out requested segment `SegDescrip` subdirectories and write them to `/tmp/Segs`. The segments are to be packaged and placed in subdirectory `/ftp/pub` for transmission by `COERmtInstall`. The proper `MakeInstall` command is (this is all intended to be on one line)

```
MakeInstall -Cd COESegDescrip /tmp/Segs -Co COESeg /h/tmp
-p /home3/local SEWC ASW -so IDE RDA -p /ftp/pub -o
mysegs
```

C.2.7 mkSubmitTar

Segments must be packaged by compressing and encrypting them prior to submitting them electronically to CSRS. `mkSubmitTar` does this task. It checks to see if `VerifySeg` has been run successfully on the segment and aborts with an error if it has not. `mkSubmitTar` also ensures that the directories defined in Chapter 5 as required for segment submission are provided.

`mkSubmitTar` allows segments to be checked out of a repository prior to packaging for submission. The command line parameters `-h`, `-p`, and `-V` are supported. If `-p` is not specified, the implied path is `/h`. The following command line parameters are also supported:

`-Co exe path` — *exe* is the name of an executable that will check segments out of the repository and place them in the temporary directory indicated by the absolute pathname, *path*. The location of the executable is determined by the most recent occurrence of the `-p` parameter.

`-d on | off` — This parameter specifies whether to delete (`-d on`) or not to delete (`-d off`) segments checked out of the repository when `mkSubmitTar` completes. By default, segments are not deleted from the temporary directory.

`-o path` — *path* is the place to put the tar file created by `mkSubmitTar`. If *path* is not specified, the default is the same directory that contains the segment.

The `status` environment variable is set to -1 if an error occurs and to 0 otherwise.

`mkSubmitTar` will only package one segment into a tar file. Each component of an aggregate is packaged separately. However, `submit` allows multiple files to be sent during one session. An entire aggregate should be sent at one time unless the size is prohibitive.

C.2.8 MoveSeg

`MoveSeg` allows a segment that has already been installed to be moved from one location to another, including to another disk partition. The syntax is

```
MoveSeg {parameters} segdir path
```

where *segdir* is the present segment location and *path* is the absolute pathname for the desired new location. An error will be generated if pathname does not exist, or if there is not enough room to move the segment. The old segment location (*segdir*) is deleted when the move is completed.

`MoveSeg` accepts command line parameters `-h`, `-p`, `-v`, `-V`, and `-w`. The `status` environment variable is set to -1 if any error occurs, otherwise to 0.

For example, move the segment `MySeg` located in directory `/home2/test` to `/home3/passed/MySeg`. The proper command line is

```
MoveSeg -p /home2/test MySeg /home2/passed
```

C.2.9 SegCatalog

This tool adds a segment to the segment catalog. The syntax for the command is

```
SegCatalog {parameters}
```

Parameters supported are `-h`, `-p`, and `-V` in addition to

`-s {segs}` *segs* is a list of one or more segments to add to the catalog. The location of the segment is determined by the most recent occurrence of the `-p` parameter.

`-S {files}` *files* is a list of one or more files that contain a list of segments to add to the catalog. The location of the segments is determined by the most recent occurrence of the `-p` parameter, or as determined by `-p` parameters *within* the file.

C.2.10 submit

This tool electronically transmits segments packaged by `mkSubmitTar` to the host repository. The format of the command is

```
submit {parameters}
```

Command line parameters `-C`, `-h`, `-p`, `-v`, `-V`, and `-w` are supported in addition to the following:

`-d on | off` This parameter specifies whether to delete (`-d on`) or not to delete (`-d off`) segments after they have been transmitted. By default, segments are *not* deleted after transmission.

`-s {segs}` *segs* is a list of one or more segment files created by `mkSubmitTar`.

For example, submit the segments `IDA.Z.tar` and `RDA.Z.tar` as packaged by `mkSubmitTar` from the directory `/home3/submit`. Delete the files after transmission. The proper command is

```
submit -p /home3/submit -d on -s IDA RDA
```

C.2.11 TestInstall

`TestInstall` is used to temporarily install a segment that already resides on disk. It must be run when there are no other COE processes running. The reason for this restriction is that the tool may modify COE files already in use with unpredictable results. `VerifySeg` must have been run before `TestInstall` to make sure that the segment is valid.

`TestInstall` performs the same operations as `COEInstaller` except that it does not need to read the segment from tape (e.g., it is already on disk), and the segment may be in any arbitrary location. `TestInstall` will establish the required symbolic link under `/h` to preserve the COE standard directory structure.

Command line parameters supported are `-C`, `-h`, `-p`, `-v`, `-V`, and `-w` in addition to the following:

`-e`	Echo the descriptor contents as they are processed. Enabling this option also enables the `-f` option.
`-f`	Echo the descriptor names as they are processed.

The `status` environment variable is set to 0 if the installation is successful, and to -1 otherwise. Error and status messages are written to `stdout` as required.

C.2.12 TestRemove

`TestRemove` is used to remove a segment that was installed by `TestInstall`. It must be run when there are no other COE processes running. The reason for this restriction is that the tool may modify COE files already in use with unpredictable results. `TestRemove` removes the symbolic link under `/h` if one exists, but it does *not* delete the segment from disk.

Command line parameters supported are `-C`, `-h`, `-p`, `-v`, `-V`, and `-w` in addition to the following:

`-e`	Echo the descriptor contents as they are processed. Enabling this option also enables the `-f` option.
`-f`	Echo the descriptor names as they are processed.

The `status` environment variable is set to 0 if the removal is successful, and to -1 otherwise. Error and status messages are written to `stdout` as required.

Note: TestRemove is unconditional and should be used with caution. It will remove a specified segment even if other segments still installed depend upon it.

C.2.13 TimeStamp

TimeStamp puts the current time and date into the VERSION descriptor. It is intended to assist the configuration management process by allowing the time stamp to be updated just prior to running VerifySeg and mkSubmitTar for the deliverable product.

Command line parameters supported are -h, -p, and-V. The status environment variable is set to -1 if an error occurs (e.g., VERSION descriptor not found) and 0 otherwise.

C.2.14 VarAnalyze

VarAnalyze analyzes the segment specified and creates a table showing total memory and disk usage. It determines all segment dependencies so that the statistics reported include all required segments.

Command line parameters supported are -h, -p, and -V. The status environment variable is set to -1 if an error occurs (e.g., segment not found, dependent segment not loaded) and 0 otherwise.

C.2.15 VarDef

VarDef is used to create a variant definition from a list of segments. Command line parameters supported are -C, -h, -p, -v, -V, and -w in addition to the following:

-Cd exe path	*exe* is the name of an executable that will check segment descriptors out of the repository and place them in the temporary location indicated by the absolute pathname, *path*. The location of the executable is determined by the most recent occurrence of the -p parameter. Temporary segment descriptors are automatically deleted after the segments are processed.
-f	Print segment names as they are processed.
-n name	*name* is a string containing the name to assign to the variant.

`-o varname` *varname* is the name of the file to contain the variant definition.

`-s {segs}` *segs* is a list of one or more segments to include in the variant. All components of an aggregate must be included or else an error is generated. The location of the segments is determined by the most recent occurrence of the `-p` parameter. These segments are *not* in a repository.

`-so {segs}` *segs* is a list of one or more segments to include in the variant definition. The segment descriptors are to be checked out of the repository by the program provided by the `-Cd` parameter.

`-S {files}` *files* is a list of one or more files which contain a list of segments to include in the variant definition. The location of the file is determined by the most recent occurrence of the `-p` parameter, while the segments are either located under `/h`, or as determined by `-p` parameters *within* the file. These segments are *not* in a repository.

`-So {files}` *files* is a list of one or more files which contain a list of segments whose segment descriptors are to be checked out of the repository specified by the `-Cd` parameter.

The `status` environment variable is set to 0 if successful, and to -1 if any errors occur. Error messages are written to `stdout` as required.

C.2.16 VerifyCOE

The tool `VerifyCOE` is used to validate the COE runtime environment. It is intended to be used after the kernel COE has been loaded and configured to ensure that it has been correctly set up. This tool is primarily intended for developers who wish to alter the COE installation instructions to conform to development environment needs.

To execute the tool, perform the following steps:

1. Perform a clean reboot of the computer.

2. Log in as `root`.

3. Confirm that no system resources, such as shared memory, are already allocated or in use.

4. Confirm that no COE background processes are running.

5. Use the Unix `source` command to execute any scripts needed to establish the environment that is to be tested.

6. Execute the `VerifyCOE` program.

The format of the command is

```
VerifyCOE {parameters}
```

where the supported command line parameters are `-h`, `-v`, `-V`, and `-w`. All output is written to `stdout`.

Specific tests performed include:

- Check for sufficient swap space on the disk.
- Check for correct disk partitioning.
- Check for expected device drivers.
- Check for sufficient shared memory, message pool size, and other operating system kernel parameters.
- Check for proper location of X and Motif libraries.

C.2.17 VerifySeg

`VerifySeg` is used to validate that a segment conforms to the rules for defining a segment. It uses information in the `SegDescrip` subdirectory and must be run whenever the segment is modified. Command line parameters supported are `-C`, `-h`, `-p`, `-R`, `-v`, `-V`, and `-w` in addition to the following:

`-e`	Echo segment descriptor lines as they are processed (also includes the `-f` option)
`-f`	Print descriptor names as they are processed
`-o`	Process the segment and identify obsolete usage (e.g., `ModName` vs. `SegName`)
`-s name`	Validate just the single descriptor *name*

`-t` Print a table showing required / optional descriptors for each segment type

`-x [name]` Display the syntax for the named descriptor *name*, or all descriptors if *name* is not specified.

Run the `VerifySeg` tool for each segment. If the segment is an aggregate segment, `VerifySeg` must be run on each segment in the aggregate.

The `status` environment variable is set to -1 if any errors were encountered, and to 0 otherwise. Error and status messages are written to `stdout` as required. Specific tests include:

- ¥ Check to ensure that executables are prefaced with the segment prefix where required.
- ¥ Check to ensure that all defined environment variables use the segment prefix.
- ¥ Check to ensure that all pathnames are defined relative to the home environment variable, `segprefix_HOME.`
- ¥ Check to ensure that all required environment variables (`USER_HOME`, `USER_DATA`, `DATA_DIR`, etc.) are defined by account group segments.
- ¥ Check to ensure that all required environment variables are defined by the COE parent component segment.
- ¥ Check aggregate segments for internal consistency (e.g., all components are hardware compatible).
- ¥ Check and warn if `mv` is used in a `PostInstall`, `PreInstall`, or `DEINSTALL` script since this may be an illegal attempt to move files across disk partitions.

C.2.18 VerUpdate

`VerUpdate` updates the segment version number, date, and time in the `VERSION` descriptor. Command line parameters supported are `-h`, `-p`, and `-V` in addition to the following:

`-i version` *version* is the version number to insert unconditionally.

`-d digit` Increment the digit specified by *digit* in the version number. *digit* is 1 for the major release digit, 2 for the minor release digit, 3 for the maintenance release digit, or 4 for the developer digit.

If no version number is specified, the version number inside the `VERSION` descriptor is incremented. That is, 1.0.0.0 is updated to be 1.0.0.1. If no version is specified, and `VERSION` does not exist or has no version number, the version number 1.0.0.0 is inserted.

The command

```
VerUpdate -d 2
```

would increment the minor release number so that the version number 2.1.0.5 would be changed to 2.2.0.5. Multiple digits may be specified at the same time so that

```
VerUpdate -d 234
```

would change the same version number to 2.2.1.6.

`status` is set to 0 if successful, and to -1 if an error occurs (e.g., segment not found).

> **Note:** No error checking is performed on the version number specified since the segment must still be validated by `VerifySeg`.